Copyright (C) 1989, 1990, 1991 Aladdin Enterprises. All rights reserved.
Distributed by Free Software Foundation, Inc.

This file is part of Ghostscript.

Ghostscript is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY. No author or distributor accepts responsibility
to anyone for the consequences of using it or for whether it serves any
particular purpose or works at all, unless he says so in writing. Refer
to the Ghostscript General Public License for full details.

Everyone is granted permission to copy, modify and redistribute
Ghostscript, but only under the conditions described in the Ghostscript
General Public License. A copy of this license is supposed to have been
given to you along with Ghostscript so you can know your rights and
responsibilities. It should be in a file named COPYING. Among other
things, the copyright notice and this notice must be preserved on all
copies.

- -

This file, use.doc, describes how to use the Ghostscript language
interpreter.

For an overview of Ghostscript and a list of the documentation files, see
README.

****
******** How to use the Ghostscript interpreter ********
****

The file name of the Ghostscript interpreter is gs.exe (MS-DOS and VMS) or
gs (Unix). To run it, you also need the initialization files gs_init.ps,
gs_fonts.ps, gs_statd.ps, and symbol_e.ps; if you are going to be
displaying any text, you also need uglyr.gsf and Fontmap.

To invoke the interpreter, give the command
        gs <filename1> ... <filenameN>
The interpreter will read in the files in sequence and execute them.
After doing this, it reads further input from the primary input stream
(normally the keyboard). Each line (i.e. characters up to a <return>) is
interpreted separately. To exit from the interpreter, type quit<return>.
The interpreter also exits gracefully if it encounters end-of-file.
Typing the interrupt character, e.g., control-C, is also safe.

The interpreter recognizes several switches described below, which may appear
anywhere in the command line and apply to all files thereafter.

Ghostscript may be built with multiple output devices. Ghostscript
normally opens the first one and directs output to it. To use device xyz
as the initial output device, include the switch
        -sDEVICE=xyz
in the command line. Note that this switch must precede the first .ps
file, and only its first invocation has any effect. For example, for
printer output in a normal configuration that includes an Epson printer
driver, you might use the shell command

```
        gs -sDEVICE=epson myfile.ps
```
instead of just
```
        gs myfile.ps
```
Alternatively, you can type
```
        (epson) selectdevice
        (myfile.ps) run
```
All output then goes to the printer instead of the display until further notice. You can switch devices at any time by using the selectdevice procedure, e.g.,
```
        (vga) selectdevice
```
or
```
        (epson) selectdevice
```

To find out what devices are available, just type
```
        devicenames
```
The -h and -? (help) message also lists the available devices.

File searching

When looking for the initialization files (gs_*.ps), the files related to fonts, or the file for the 'run' operator, Ghostscript first tries opening the file with the name as given (i.e., using the current working directory if none is specified). If this fails, and the file name doesn't specify an explicit directory or drive (i.e., doesn't begin with '/' on Unix systems; doesn't contain a ':' or begin with a '/' or '\' on MS-DOS systems; doesn't contain a ':' or a square bracket on VMS systems), Ghostscript will try directories in the following order:

- The directory/ies specified by the -I switch(es) in the command line (see below), if any;

- The directory/ies specified by the GS_LIB environment variable, if any;

- The directory/ies specified by the GS_LIB_DEFAULT macro in the Ghostscript makefile, if any.

Each of these (GS_LIB_DEFAULT, GS_LIB, and -I parameter) may be either a single directory, or a list of directories separated by a character appropriate for the operating system (':' on Unix systems, ';' on VMS systems, ';' on MS-DOS systems).

VMS-specific notes

On VMS systems, the last character of each "directory" name indicates what sort of entity the "directory" references. If the "directory" name ends with a colon, it is taken as referring to a logical device, e.g.:
```
    $ DEFINE GHOSTSCRIPT_DEVICE DUA1:[GHOSTSCRIPT_14]
    $ DEFINE GS_LIB GHOSTSCRIPT_DEVICE:
```
If the "directory" name ends with a closing square bracket, it is taken as referring to a real directory, e.g.:
```
    $ DEFINE GS_LIB DUA1:[GHOSTSCRIPT]
```

To run Ghostscript with switches, you must type a command like

```
$ gs "-dNODISPLAY"
```

because the C run time library will convert the command parameters/arguments to lowercase unless you enclose them in double quotes which preserves the case.

If you are on an X Windows display (for which gs is built), you can do

```
$ set display/create/node="domain-name"/transport=tcpip
```

For example,

```
$ set display/create/node="doof.city.com"/transport=tcpip
```

and then run Ghostscript

```
$ gs
```

MS-DOS resources

If you are running Ghostscript on a MS-DOS machine with a display that is not EGA/VGA compatible, you must build Ghostscript with the BGI driver as the default, and you will need the appropriate .BGI file from the Borland Turbo C library. (Ghostscript includes the EGA/VGA driver in the executable.)

If you are using the BGI driver, two additional environment variables become relevant:

BGIPATH - defines the directory where Ghostscript will look for the appropriate BGI driver. If BGIPATH is not defined, Ghostscript will only look in the current directory.

BGIUSER - a string of the form nn.dname, where nn is a hexadecimal number giving a display mode and dname is the name of a file containing a user-supplied BGI driver. If BGIUSER is defined and the BGI device is selected, Ghostscript will supply nn as the display mode and will obtain the driver from the file named dname.

X Windows resources

Ghostscript looks for the following resources under the program name "Ghostscript":

```
borderWidth - the border width in pixels
        default = 1
borderColor - the name of the border color
        default = black
geometry - the window size and placement, WxH+X+Y
        default = ???
xResolution - the number of x pixels per inch
        default is computed from WidthOfScreen and WidthMMOfScreen
yResolution - the number of y pixels per inch
```

default is computed from HeightOfScreen and HeightMMOfScreen

To set these resources, put them in a file (such as ~/.Xdefaults) in the following form:

```
Ghostscript*geometry:          612x792-0+0
Ghostscript*xResolution: 72
Ghostscript*yResolution: 72
```

Then load the defaults into the X server:

```
% xrdb -merge ~/.Xdefaults
```

Normal switches

-- filename arg1 ...
Takes the next argument as a file name as usual, but takes all remaining arguments (even if they have the syntactic form of switches) and defines the name ARGUMENTS in userdict (not systemdict) as an array of those strings, *before* running the file. When Ghostscript finishes executing the file, it exits back to the shell.

-Dname=token
-dname=token
Define a name in systemdict with the given definition. The token must be exactly one token (as defined by the 'token' operator) and must not contain any whitespace.

-Dname
-dname
Define a name in systemdict with value=null.

-Sname=string
-sname=string
Define a name in systemdict with a given string as value. This is different from -d. For example,

```
-dname=35
```

is equivalent to the program fragment

```
/name 35 def
```

whereas

```
-sname=35
```

is equivalent to

```
/name (35) def
```

-q
Quiet startup -- suppress normal startup messages, and also do the equivalent of -dQUIET.

-gnumber1xnumber2
Equivalent to -dDEVICEWIDTH=number1 and -dDEVICEHEIGHT=number2. This is for the benefit of devices (such as display windows) that require (or allow) width and height to be specified.

-Idirectories
Adds the designated list of directories at the head of the search path for library files.

-
This is not really a switch. It indicates to Ghostscript that the standard input is coming from a file or a pipe. Ghostscript reads from stdin until reaching end-of-file, executing it like any other file, and then continues processing the command line. At the end of the command line, Ghostscript exits rather than going into its interactive mode.

Note that gs_init.ps makes systemdict read-only, so the values of names defined with -D/d/S/s cannot be changed (although, of course, they can be superseded by definitions in userdict or other dictionaries.)

Special names

Including -dNODISPLAY in the command line will suppress the normal initialization of the output device. This may be useful when debugging. Similarly, -dNOCACHE disables character caching, and -dNOBIND disables the 'bind' operator. -dNOPAUSE disables the prompt and pause at the end of each page; this may be desirable for applications where another program is 'driving' Ghostscript.

As described above, -sDEVICE=device selects an alternate initial output device.

For the benefit of PostScript files that think they are being printed on a hardcopy printer, gs_init.ps includes the line
(gs_statd.ps) run
This file contains dummy definitions for statusdict.

Debugging switches

The -T and -Z switches only apply if the interpreter was built for a debugging configuration.

-Mn Force the interpreter's allocator to acquire additional memory in units of nK, rather than the default (currently 20K on MS-DOS systems, 50K on Unix). n is a positive decimal integer (not exceeding 63 on MS-DOS systems).

-Tname:rsize:format
(Only available on MS-DOS systems.)
Trace a procedure within Ghostscript. Any number of procedures may be traced. rsize is the size of the value returned by the procedure, in bytes; if omitted, the value returned by the procedure is not printed. format is a printf-style template for printing the arguments; if omitted, the arguments are printed in hex. Note that MS-DOS provides no way to quote spaces or commas within a command line argument.

-Zxxx Turn on debugging printout.
Each of the xxx characters selects an option:
if the string is empty, all options are selected.
Case is significant.

0 = type 1 path tracer (type1imagepath)
1 = type 1 font interpreter (type1addpath)
2 = curve subdivider
a = allocator (large blocks only)
A = allocator (all calls)
b = bitmap image processor
B = bitmap images, detail
c = color/halftone mapper
d = dictionary lookup
e = error returns
E = error returns, dump C stack
f = fill algorithm (summary)
F = fill algorithm (detail)
h = halftone renderer
i = interpreter, just names
I = interpreter, everything
k = character cache
K = character cache, every access
l = command lists, bands
L = command lists, everything
m = makefont and font cache
o = outliner (stroke)
p = path tracer
q = rectangle fill
r = arc renderer
s = scanner
t = tiling algorithm
u = undo saver (for save/restore)
v = device-level output
w = LZW encoder/decoder
x = transformations